
Towards more Realistic and Efficient Virtual Environment Description and Usage

Sébastien Paris*, Stéphane Donikian†, Nicolas Bonvalet‡

IRISA, Campus de Beaulieu, 35042 Rennes cedex - FRANCE

AREP, 163 bis, av. de Clichy - impasse Chalabre, 75017 Paris - FRANCE

email: {sebastien-guillaume.paris, stephane.donikian}@irisa.fr, nicolas.bonvalet@arep.fr

Abstract

We treat in this paper two linked subjects underlying virtual environment description: topological abstraction, and the way its precomputed data can be accessed by autonomous agents. The topological abstraction we propose is composed of three hierarchical layers, the first being the subdivision cells graph, and the two others being successive groupings. Then, we highlight some preprocesses computable with abstraction in order to improve environment information accessibility, such as potential visibility sets. Agents can then retrieve globally computed data to use them in an egocentric realistic way. Moreover, we constrained our model so that it is usable within the framework of microscopic crowd simulation. Thus, it is sufficiently realistic, very efficient in access time, and can handle small as well as large environments.

Keywords

Informed environment; Topological abstraction, observation and knowledge; Microscopic simulation.

1 Introduction

Simulation of autonomous agents is a wide and complex topic, requiring the description of all possible actions for each individual. One of the principal actions, necessary to any autonomous agent, is unquestionably the capacity to move. To achieve this goal, a path planning algorithm must be provided, as well as a virtual environment description to reason on. This study focuses on this last point by trying to answer to the following questions : how can a virtual environment be described in order to provide easy and fast accessibility to its data, and how can this description be used in a realistic way to achieve some human like behaviours? Moreover, two major constraints are imposed by the underlying purpose of this study, which is crowd simulation inside constrained environments, and particularly exchange areas like train stations. First, the proposed algorithms must be sufficiently powerful to simulate a great number of entities, in order to manage crowds of people microscopically. Second, these algorithms must be realistic enough to allow an evaluation and a validation by comparison with real situations, and to produce data which can be studied. First of all, related work, in section 2, approaches spatial

*AREP; PhD student SIAMES, IRISA

† IRISA/CNRS

‡ AREP

subdivision, which will not be detailed any further in this study. Then, section 3 presents our hierarchical topological abstraction allowing the description of conceptual circulation areas, and will highlight some preprocesses which can improve data accessibility. Section 4 shows how this abstraction can be used to simulate environment observation by an agent, and thus the acquisition and recovery of individual knowledge. Section 5 presents some results, including technical benchmarks. Finally, future work is exposed in section 6.

2 Related work

Spatial subdivision converts complex geometric data, made up of a great number of polygons, in a more or less informed database describing the environment in a more conceptual way. The first approaches come from the field of robotics [9] with three principal theories: potential fields, roadmaps, and cell decomposition. Then, behavioural animation has shown an interest in this subject, by adapting and improving these theories. *Potential fields* [4] associate repulsive powers with the environment obstacles, and an attractive one with the agent destination. *Roadmaps* [2, 10, 12] discretise navigation space in a network of paths made up of lines and curves. *Cell decomposition* can be carried out in two different ways. An approximate decomposition, by uniform or non uniform grids [7, 13], or even quadtrees [11], covers a subspace included in the environment's free space. An exact decomposition [1, 6], generally by convex cells, covers the whole of the environment's free space. In both cases, a graph is obtained whose nodes represent cells, and arcs describe connectivity relations. **Topological abstraction** is an additional process used to better organise the information obtained at the time of spatial subdivision. This topic is often approached jointly with spatial subdivision since it is a complementary process. The unification heuristic is principally approached in two ways. The first way unifies basic subdivision cells in a mathematical way [8], to obtain bigger structures. The second way unifies cells in a more conceptual way, to introduce a semantical definition of the environment, like with the IHT-graph structure [14]. Finally, **path planning** and **navigation** is performed, potentially by using an individual knowledge database for agents [5]. Many algorithms are used to plan a path inside an environment, principally based on the well known A^* algorithm. One of the notable improvements of A^* is HPA^* [3], for *Hierarchical Path-finding* A^* , which proposes to plan a path based on a hierarchical description of the environment by grids. One can notice that the environment description we propose is very well suited for an exploitation by this kind of HPA^* algorithm.

3 Topological abstraction

In order to reason about an environment, to plan a path, it is necessary to describe it. Although a purely graphical definition could be directly exploited, it is not well suited for many complex requests. Therefore, a more ordered and accessible description is needed. That's why our model starts by extracting a topological abstraction from the convex cells obtained at the time of the environment spatial subdivision. This spatial subdivision is provided by a part of the algorithm described by Lamarche et al. [8]. The algorithm is applied to a 2D map representation of the environment, and extracts a set of convex cells by computing a constrained Delaunay triangulation. Moreover, obtained cells are refined by taking into account shortest distances between corners and walls. The original algorithm then computes a hierarchical abstraction of the environment which will not be used for this study.

3.1 Informed subdivision graph

The first step of our model consists in informing the spatial subdivision (Fig. 1(a)). Thus, a connectivity graph is built (Fig. 1(b)), whose nodes represent convex cells identified according to their type: **Dead end** if the cell comprises only one connectivity relation, **Corridor** if the cell comprises exactly two connectivity relations, and **Crossroad** for other cells which comprise at least three connectivity relations. Moreover, two others pieces of information are associated to some particular cells of the environment. A **virtual** cell is connex to the environment’s virtual limits, i.e. one of its vertices belongs to the environment bounding polygon. These cells do not correspond to any real place, but could be used for navigation. Cells named V_n on figure 1 are *virtual*. An **entry/exit** cell is both connected to a virtual cell and to at least one other cell (it can’t be a dead end), without being virtual itself. These cells should be considered as the real simulation limits. Both cells A and O on figure 1 are *entries/exits*.

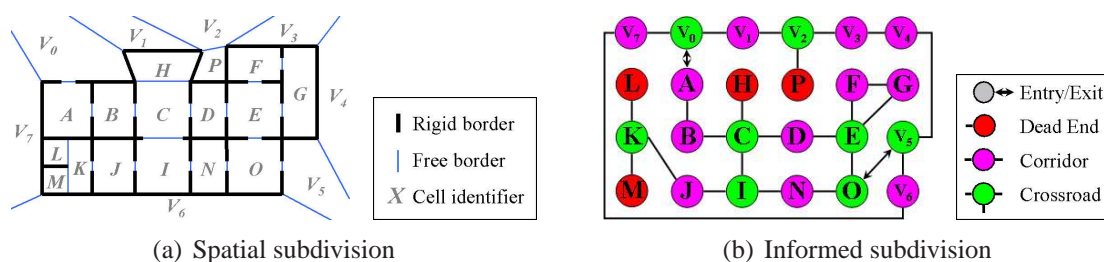


Figure 1: Example of the informed subdivision process for a simplified environment

The example in figure 1 is relatively simple, as it only contains twenty four cells. But, for more complex cases, the informed subdivision graph could contain a very large amount of cells, up to several thousands. The reason comes from the fact that the subdivision produces convex cells, which is very sensitive to the layout imperfections, such as on environments with curved limits. Even if path planning can be performed with such a large abstraction, we would encounter performance problems very quickly when a great number of requests becomes necessary, as is the case for crowd simulation. A solution consists in reducing the search space, by successive groupings which drop the convexity constraint.

3.2 Hierarchical abstraction

In order to reduce the search space, a hierarchy of abstracted graphs is proposed by grouping connex zones. Each node of a graph of the $n + 1$ level contains a subset of the graph of level n , composed by at least one node. The extraction of these new graphs imposes only a low memory overhead. Indeed, each grouping phase drastically reduces the number of nodes. On the other hand, two points led us to limit the number of abstraction phases. **Expressivity** decreases with each grouping, which could bring to useless additional planning stages because of their low discrimination potential. Indeed, the more an abstract node contains potential sub paths, the less its choice impacts planning. **Traversal cost** increases with each grouping, which could reduce performance gain brought by a hierarchical representation. Indeed, whatever the number of levels of the hierarchical graph, the planning process must provide the list of basic cells to pass through. Thus, the hierarchical graph must inevitably be traversed from top to bottom to reach the informed subdivision level. That’s why our model only contains two abstraction phases, which provide a three level hierarchical graph. The abstractions number limitation has two consequences. On the one hand, it gets an incomplete abstraction in some cases, which is not very annoying since the cell number of the most abstracted graph remain within acceptable

limits. On the other hand, it produces a constant number of three hierarchical graphs allowing to perform an effective path planning by parts.

3.2.1 First abstraction stage

The purpose of the first abstraction stage is to filter subdivision granularity, like multiple convex cells for a curved corridor (Fig. 2(a)), or dead end emergence for little recesses (Fig. 2(b)).



Figure 2: Examples of subdivision granularity

So, the goal here is to obtain a graph describing the environment in a more conceptual way. Let us name the nodes of this graph **groups**. They are computed using the following three step recursive algorithm :

- 1) Group all **virtual** cells into a virtual crossroad group, then:
 - Absorb all dead end cells connected to a cell of this group.
 - If the virtual group is empty, proceed to step 3 on any crossroad cell.
 - Otherwise, proceed to step 2 or 3 according to the type of any connex ungrouped cells, which are the environment **entries/exits**.

This first step produces a unique virtual group representing the environment *outside*. This place can be used for agents generation, but should be avoided while path planning.

- 2) All ungrouped connex **corridor** cells are grouped in a corridor group, then:
 - If this corridor group is connex to a dead end cell, absorb it and transform the corridor group into a dead end.
 - Proceed to step 3 on connex ungrouped crossroad cell if any.
 - If this corridor group is connected twice to the same crossroad group, merge it with that group.

This second step removes very discretized linear structures (Fig. 2(a)), which are often responsible for useless calculations. Moreover, it highlights the first global structures being able to be quickly refuted during destination oriented path planning.

- 3) All ungrouped connex **crossroad** and **dead end** cells are grouped in a crossroad, then:
 - Proceed to step 2 on any connex ungrouped corridor cells.
 - If the crossroad group has only one connectivity relation, merge it with the adjacent corridor group inside a dead end.
 - If the crossroad group has only two connectivity relations, merge it with both adjacent dead end or corridor groups inside a dead end (if one of the merged groups is a dead end) or a corridor (then apply step 2 checks).

This third step reduces the number of connectivity relations that crossroads are responsible for. It produces one big crossroad with many connectivity relations, instead of many crossroads comprising the same number of connexities added to interconnections. Moreover, it transforms false crossroad cells produced by subdivision, which are only connected to one or two corridors and to many dead ends (Fig. 2(b)).

Once these operations are finished, two inter-connected graphs are obtained. As one can see on figure 3(a), the grouping enables us to considerably reduce the number of nodes (empirically by two thirds). A property of the obtained graph is that corridor and dead end groups are inevitably connected to crossroads, but never to another corridor or dead end. Crossroad groups follow the same rule, not being able to be connected between them without the intermediary of a corridor, except for the case of entry/exit crossroad groups which are connected to the virtual one. Even if this graph allows more efficient searches than the original, it still depends too much on the subdivision method.

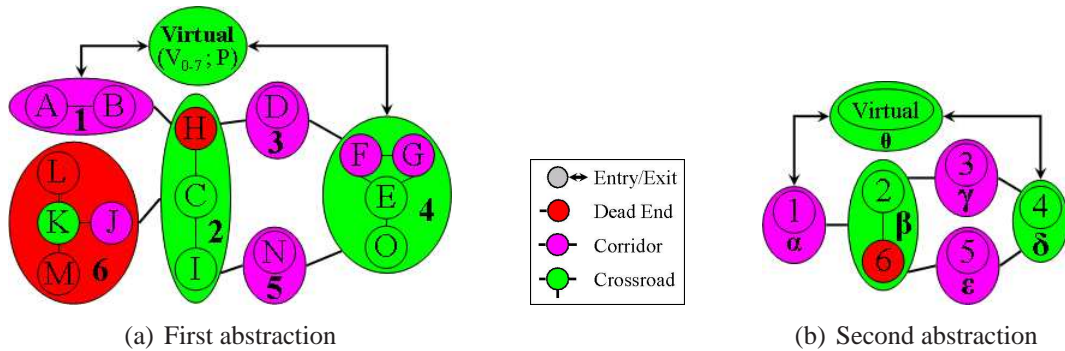


Figure 3: Abstraction processes for the informed subdivision example

3.2.2 Second abstraction stage

The second, and last, abstraction stage reduces the subdivision dependency of the graph structure. Let us name the nodes of this graph *zones*. The same grouping algorithm as for the first stage is used, working on groups and producing interconnected zones (Fig. 3(b)). *Step 1* makes the virtual zone absorb entry/exit dead end groups, which are of little use during path planning. *Step 2* becomes useless because of the connectivity properties of the source graph, except for the check part which may be used by the following step. *Step 3* is the most significant, making crossroad zones absorb dead end groups, and eventually be transformed into corridors or dead ends. Indeed, the cost of path planning calculation heavily depends on the number of crossroads, which are the only areas requiring a choice. The zones graph has the same properties as the groups graph in terms of zones connectivity types. As one can see in figure 4(c), it permits to reduce even more the number of nodes, and obtain a much more conceptual subdivision.

3.3 Informed abstraction

The first goal of the topological abstraction is to give an environment description which can be easily and quickly accessed. With the hierarchical graph, an accessible topological structure is provided, but some information is missing to qualify the graph nodes for a path planning process.

Groups and zones surfaces can be used to approximate densities of people. As we'll see later, a density criterion can be computed by first using an approximation provided by groups and zones, then by refining this evaluation when local densities are needed.

Groups and zones borders distances can be used to quickly evaluate a shortest path criterion. Since a grouping node can contain several paths between its borders, in particular for crossroads aggregates, a choice is necessary to handle precalculation. The shortest path is a good representative value, allowing to compare zones between each other. So, the shortest path is evaluated for each group and zone between all of their borders.

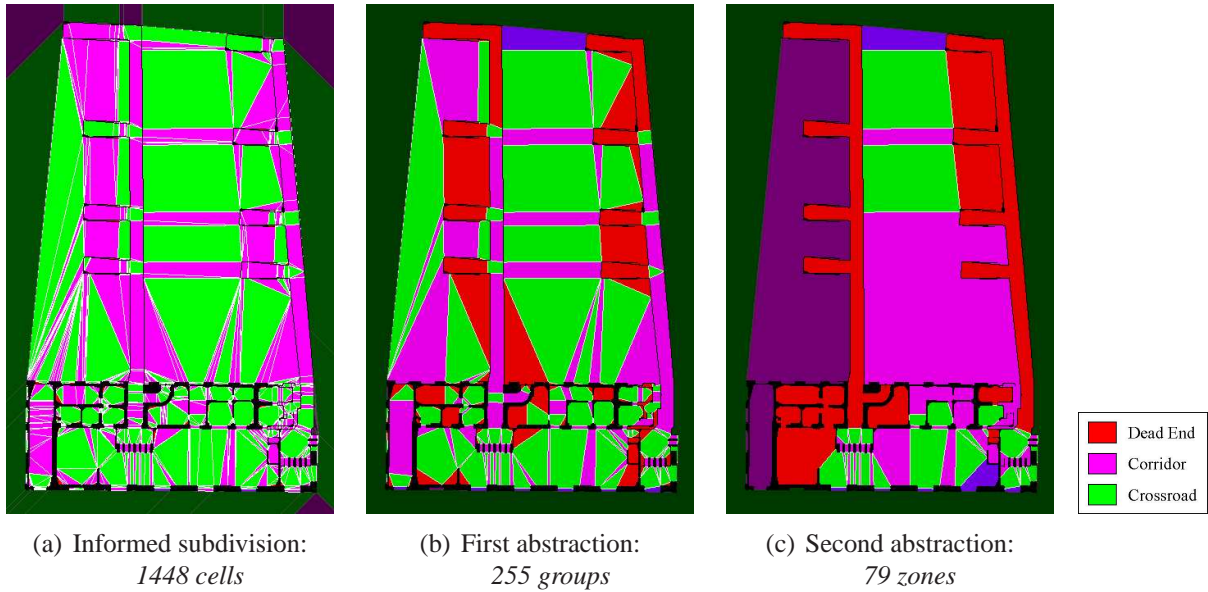


Figure 4: Full abstraction processes of a test environment

Potential visibility sets (PVS) optimise the perception process. Of course, it is impossible to precalculate all visible zones from all possible positions in the environment following all angles of sight. So, some representative positions have to be chosen. Connectivity borders between two nodes of the graph are good starting positions since they allow to construct a visibility path. But, the informed subdivision graph contains a very large amount of cells, possibly with a very small surface. So, if PVS are provided for this graph layer, they would represent such a quantity of information comprising such a strong redundancy that they would become unusable. That's why our model provides PVS for the first abstraction level, i.e. for groups connective borders. Moreover, one can notice that the connections of a level of a graph contain all the connections of a greater level. So, calculated PVS for the first abstraction layer cover the entire set of PVS of the second abstraction layer. PVS are provided as trees, the root of which is the starting border and the nodes the visible borders. These trees are obtained thanks to a recursive algorithm applied twice to each selected border, to create PVS on each side of the border. Figure 5(a) shows a representation of the view frustum used to calculate PVS for a test environment entry. An observation based representation of groups has been added in figure 5(b) to show the underlying connectivity of the tree.

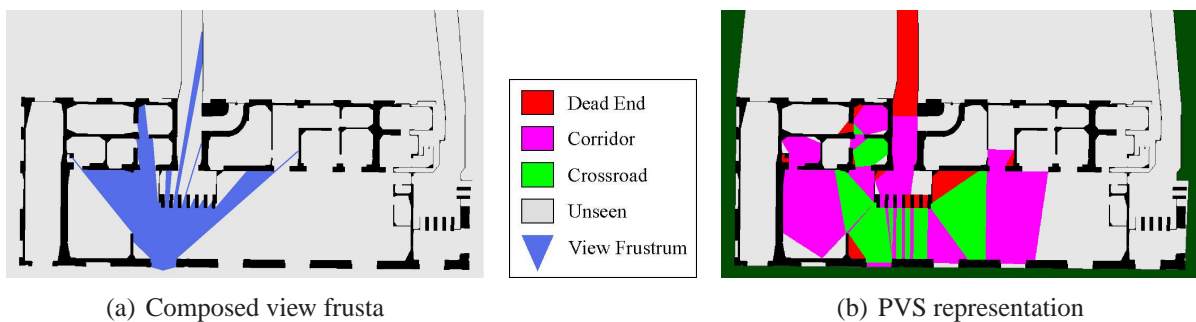


Figure 5: Precalculated visibility through an entry in a test environment.

The algorithm is initiated with the frustum through the selected border, which starting point is located on the mediator of the border opposite to the looking direction. Then, it computes composed frusta between the current one and the one defined by each border of the connex

group. If the obtained frustum is not null, i.e. it has a positive aperture, then the seen border is added to the tree children and the algorithm is recursively called. This technique thus produces an approximation of the visibility potential for each group in the environment, making it possible to carry out fast visibility tests from a given location. Indeed, the traversal of the visibility tree enables to refute the whole downward nodes of a nonvisible one, which greatly improve visibility tests through large crossroad groups with many connectivity relations.

4 Retrieving the environment information

Simulating autonomous agents behaviours implies to retrieve some information from the environment. In fact, to implement a sufficiently realistic path planning algorithm, many data should be taken into account, like densities of populations, flows of people, or even individual knowledges. We have presented in section 3 a well organised description of an environment, which makes it certainly possible to access many data, but which does not provide the recovery mechanism. Indeed, since simulated agents must be realistic, it is not possible to get any information at the time it is needed, which would lead to the simulation of omniscient beings. That is why we propose a more individual way to retrieve environment information, available thanks to the abstraction, and to interpret it.

4.1 Observation

The first method to retrieve environment information is based on individual observation. The algorithm traverses the PVS trees precalculated for each edge of the current group, starting with the agent's view frustum. For each node of the tree, the frustum is intersected with the ones defined by the agent's current position and the node children representing potentially visible borders. Then, the algorithm is recursively called on selected children. When the algorithm reaches a node, it signifies that the corresponding group is visible, even partially. Then, some additional processes can be performed depending on the kind of information the accessibility is tested for. For topological information, like group connectivity, no further processing is necessary. Indeed, each selected child of the current node gives a connectivity relation visible by the agent. For more localised information, like while searching for a specific entity, a final visibility test must be performed with the frustum obtained for the group containing the entity. Let us see two examples of information which can be observed.

Flow of people estimation is a dynamic information evaluated by a global process for each group, and accessed individually for visible groups. The global evaluation takes place as two counters associated to each group borders (one counter for each border side). A counter is incremented by each agent whose next planning step makes it cross the associated border from a side, and decremented if the agent's planning step is modified. So, if a border is considered visible by the agent, its counters become readable.

Density of people estimation can be first approximated for each group, on the same way as for flows. A counter is associated to groups, incremented by entering agents, and decremented by leaving ones. Then, if a group is considered visible, even partially, this counter becomes readable. But this estimation is hard to take into account because of the strong heterogeneity of the groups' surfaces. For example, a large group can have a globally low density with a great one concentrated on a border, making it uncrossable. So, an additional process must be applied to handle more localised densities. Such a process uses the obtained visibility region, but also needs a more precise agents localisation. To do so, another representation of the environment is

needed: a regular grid. Each box of the grid must have a sufficient surface to make it possible to quantify a rather broad panel of densities, so we chose boxes of four square meters. Boxes are correlated, during preprocessing, with groups, a box being able to belong to several groups. Then, a path is searched, to reach the group selected border, inside the grid boxes of the group belonging to the agent's view frustum. The algorithm used is a standard A^* minimising path density and length. Once the path is computed, its greatest density is used to represent the group's density (Fig. 6).

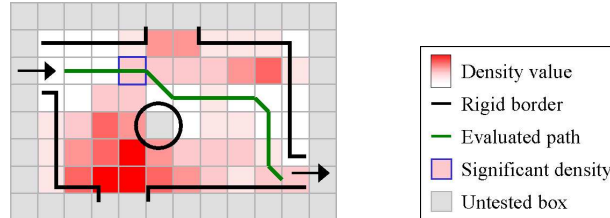


Figure 6: Example of local density estimation

4.2 Individual knowledge

The other method to retrieve environment information is by remembering it. So, we propose to simulate an agent's environment knowledge with a learning method. Let us see the three kinds of information which can be remembered.

Topological knowledge makes it possible to plan a path only through known groups of the environment, or to influence the choice of the groups to be explored. We do not use the term *cognitive map* here because the mental topological representation which is built for each agent is not in any way modified by perception phenomena or subjective interpretation. Indeed, for reasons of calculation speed and memory cost, it was decided to map the agents' mental chart to the global topological abstraction. Moreover, as the informed subdivision graph is not conceptual enough, and contains a great number of cells, we chose to represent the topological knowledge with the first abstraction graph. The topological knowledge is thus represented by an equivalence table, filled by the observation process, giving the agent knowledge about each group. Then, the knowledge of the environment zones can be extrapolated from the group data. Three levels of knowledge are managed. **Perfect**: all of the group accesses have been seen. **Partial**: some of the group accesses have been seen, they are listed. Only these accesses can be used for path planning. **Unknown**: the group is not known at all, and must not be taken into account for path planning. One can notice that this kind of knowledge enables a specific path planning computation: *exploration*. In fact, the goal of this navigation behaviour is not to reach a known location, but to increase the agent's topological knowledge by exploring the part of the environment which is not well known.

Temporal knowledge makes it possible to evaluate the validity of a dynamic information. Here the date on which a group has been seen for the last time is stored. When a dynamic information related to a group is needed, like flows of people or globally evaluated densities, an agent can test if it has seen the group a little while ago and then use the global approximation, more or less deteriorated.

Corrective locally evaluated densities are stored with a timestamp corresponding to the time they were computed. So, when a precise density information is needed, the stored value validity is tested and used if enabled. When the value is estimated as invalid, i.e. too much time has passed since its recording, it is refreshed if the corresponding group is still visible, or dropped.

5 Results

Our model has been tested with different kinds of environments, from a train station to part of a city. For the topological abstraction, the discriminating value for performances concerns memory costs rather than computation time. Indeed, this topic is a part of the simulation initialisation process, as it can be precalculated. For the agent’s topological knowledge, memory cost is also the determining value, since we need to simultaneously manage a great number of entities. The chosen test environment is a part of the city centre of Rennes, covering an area of approximately 1 km². The total abstraction process, including all precalculations like PVS, is obtained in 2 seconds with a total memory size of 35 MB. The number of nodes for each graph is : 4486 cells (*Fig. 7(a)*), 1516 groups (*Fig. 7(b)*), and 929 zones (*Fig. 7(c)*). An agent’s memory cost for this environment, including its topological knowledge, is 7 KB. Such a low cost enables to store almost 15.000 agents by only using 100 MB of memory. To test information access speed, we have implemented a simple *A** path planning algorithm, working by part on each layer of the abstraction graph. We added a simple navigation process to handle observation calculation through the path, and topological knowledge updates. By running a set of benchmarks by simulating one agent, with three simulation frequencies (1, 10, and 100 Hz), and with different topological knowledges (from unknown to perfect), a relatively constant simulation step duration of 0.1 ms is obtained (without graphical output). Demonstration videos can be seen at <http://www.irisa.fr/siames/paris/VCrowd05>

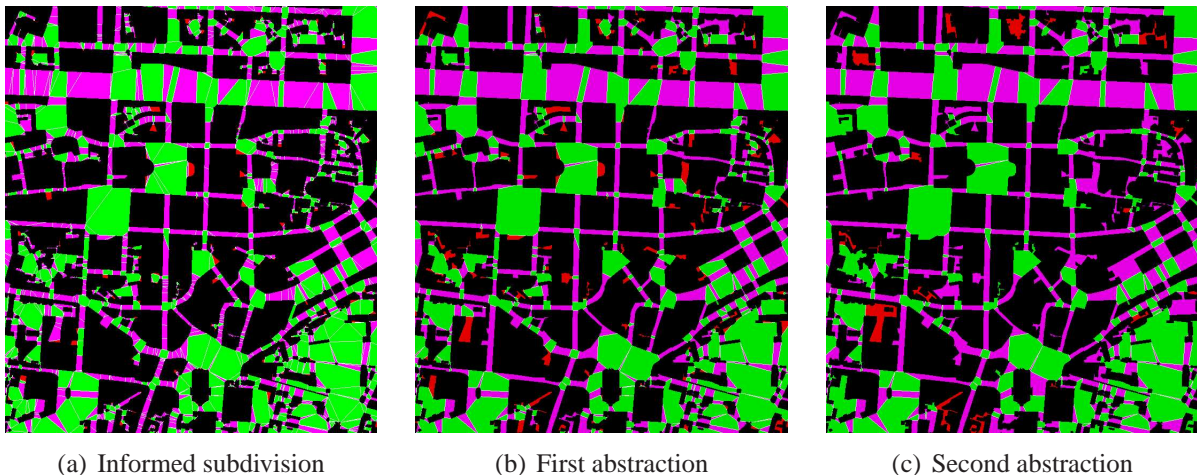


Figure 7: Full abstraction processes of the city center of Rennes

6 Conclusion and future work

The approach presented in this paper first enables to automatically describe large and complex environments to handle fast and frequent information access requests. The provided technique is an automated process only based on a graphical definition of a virtual environment. A three level hierarchical graph is built from the environment subdivision into convex cells provided by an external process. Then, this graph is informed with some precalculated data, which can be used by agents to quantify the traversal cost of a graph node. The second point describes the way to access topological information in a realistic manner. To do so, an environment observation procedure is provided, independently used by each agent. From the observation, each agent can update its own environment knowledge, which can be used either if the observation is not available anymore or if the information has been recorded a little while ago. This tool is intended to be used to characterise levels of service for exchange areas, and particularly for train stations.

Future work will treat 4 topics. First, the environment abstraction must be improved to not only manage topological nodes but also conceptual ones. So, some specific navigation nodes will be introduced, representing doors, stairs, or even windows, which could be specifically taken into account for path planning purpose or for PVS calculation. Moreover, the lists of interactive objects contained in each group could be made, making it possible to plan a path to search for specific objects. The second point to be treated is the use of the hierarchical abstraction graph by a path planning algorithm. This one should be based on a multicriteria heuristic to handle all the information available thanks to the environment observation and the agent's knowledge. The third point concerns the integration of a navigation process which must be able to manage the agent observation. Finally, as agents can actually retrieve information by observing the environment, some specific objects should be managed to give them more conceptual data. These objects could be maps or signs which will offer visual interactions to agents, allowing them to improve their environmental knowledge. This concept could then be extended to some social interaction where agents may communicate part of their knowledge to others.

References

- [1] C. Andújar, P. Vázquez, and F. M. Way-finder: guided tours through complex walkthrough models. *Computer Graphics Forum, Eurographics'04*, 2004.
- [2] O. Arikan, S. Chenney, and D. A. Forsyth. Efficient multi-agent path planning. In *Computer Animation and Simulation '01*, pages 151–162. Springer-Verlag, 2001.
- [3] A. Botea, M. Müller, and J. Schaeffer. Near optimal hierarchical path-finding. *Journal of Game Development*, volume 1, 2004.
- [4] C. Gloor, P. Stucki, and K. Nagel. Hybrid techniques for pedestrian simulations. In *4th Swiss Transport Research Conference*, Monte Verità, Ascona, 2004.
- [5] H. Hochmair. Adapting one's mental model: An essential process for successful navigation in an environment. In P. Halls, editor, *Spatial Information in the Environment, Innovations in GIS 8*, pages 147–163. Taylor Francis, 2001.
- [6] M. Kallmann, H. Bieri, and D. Thalmann. Fully dynamic constrained delaunay triangulations. *Geometric Modelling for Scientific Visualization*, 2003.
- [7] J. J. Kuffner. Goal-directed navigation for animated characters using real-time path planning and control. *Lecture Notes in Computer Science*, 1537:171–179, 1998.
- [8] F. Lamarche and S. Donikian. Crowds of virtual humans : a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum, Eurographics'04*, 2004.
- [9] J.-C. Latombe. *Robot Motion Planning*. Boston: Kluwer Academic Publishers, Boston, 1991.
- [10] M. H. Overmars. Recent developments in motion planning. In *International Conference on Computational Science (3)*, pages 3–13, 2002.
- [11] W. Shao and D. Terzopoulos. Environmental modeling for autonomous virtual pedestrians. *Digital Human Modeling for Design and Engineering Symposium*, jun 2005.
- [12] M. Sung, L. Kovar, and M. Gleicher. Fast and accurate goal-directed motion synthesis for crowds. In K. Anjyo and P. Faloutsos, editors, *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 291–300, 2005.
- [13] F. Tecchia and Y. Chrysanthou. Real time rendering of densely populated urban environments. In *RenderingTechniques '00 (10th EurographicsWorkshop on Rendering)*, pages 45–56, Brno, Czech Republic, 2000. Springer-Verlag.
- [14] R. Thomas and S. Donikian. A model of hierarchical cognitive map and human memory designed for reactive and planned navigation. In *4th International Space Syntax Symposium*, Londres, 2003.